A MODEL FOR A SPACE SHUTTLE SAFING AND FAILURE-DETECTION EXPERT

Daphna Zeilingold and John Hoey

Rockwell International Corporation
Space Transportation Systems Division
12214 Lakewood Blvd., Downey, CA 90241

ABSTRACT

The safing and failure-detection expert (SAFE) is a prototype for a malfunction detection, diagnosis, and safing system for the atmospheric revitalization subsystem (ARS) in the Space Shuttle orbiter. SAFE, whose knowledge was extracted from expert-provided heuristics and documented procedures, automatically manages all phases of failure handling: detection, diagnosis, testing procedures, and recovery instructions. The SAFE architecture allows it to handle correctly sensor failures and multiple malfunctions. Since SAFE is highly interactive, it was used as a test bed for the evaluation of various advanced human-computer interface (HCI) techniques. The use of such expert systems in the next generation of space vehicles would increase their reliability and autonomy to levels not achievable before.

INTRODUCTION

The fault detection, isolation, and recovery (FDIR) process on board the orbiter is lengthy and tedious (Figure 1). The crew is surrounded by diverse inputs: gauges, displays, warning lights, alarms, caution and warning messages, and ground control communications. If an anomaly occurs, the crew must follow flow chart-like malfunction procedures (essentially fault trees) to isolate the problem and reconfigure the vehicle to a safe state. The crew must locate the appropriate malfunction procedures in a voluminous manual, then find the cockpit switches that must be manipulated—and do it all in a timely manner in an emergency.

Automation of this process clearly would have many benefits. It would speed up fault isolation and make it more consistent and reliable; it would reduce the crew's work load and alleviate the loss-of-efficiency problem on longer missions; and it could, as confidence in the system is gained, reduce training and ground support requirements.

Beyond its immediate benefits, such a system would introduce into space vehicle management a new technology with the potential for substantially increasing vehicle autonomy—autonomy that will be essential when space flights such as the Mars mission are undertaken. Because of the communication time lag, complete responsibility for vehicle "health" will have to be assumed by on-board functions, and mission length and complexity will make automation mandatory.

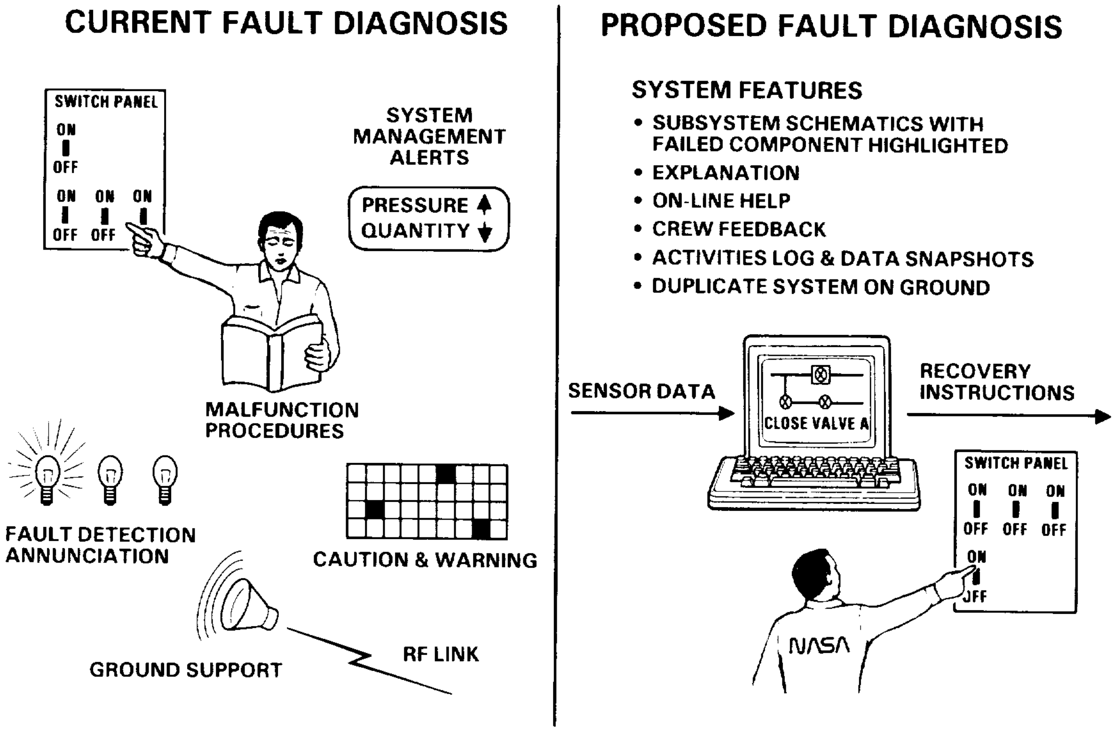

Figure 1. Comparison of Current and Proposed Fault Diagnosis

The SAFE project's objective was twofold: to investigate the use of expert system technology for on-board malfunction diagnosis through the development and prototyping of the software architecture; to experiment with various user interface techniques that would provide easy and effective crew interaction.

SAFE ARCHITECTURE

The paradigm used by SAFE for diagnosis is the "shallow-reasoning" expert system, which employs expert heuristics that map sensor signatures to component malfunctions. Its advantages are that it is fast, conceptually easy to understand, and lends itself efficiently to a rule-based structure. It reflects the way diagnosis is now done: a mixture of malfunction procedures and the expertise of the subsystem people on the ground who advise the crew. A disadvantage, which requires further research, is that such a system cannot detect an unanticipated failure, since it has no understanding on any level of how the modeled system works. (An attempt to do diagnosis with a model-based paradigm was deferred because of severe performance problems.)

As shown in Figure 2, components of the SAFE system are the knowledge base (rules and facts), the user interface (UI), and the ARS simulator (SIM) and the simulator interface (SI). The inference engine is OPS83, which is a forward chainer—i.e., data-driven. The user interface is written in Ada and communicates with OPS83 via VAX mailboxes.

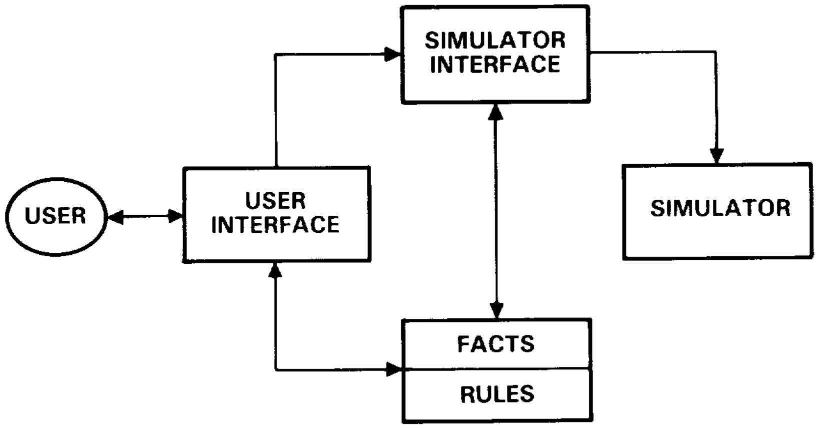

Figure 2. SAFE Overall Architecture

ARS Model

The system model used by SAFE represents a simplified view of the ARS of the Shuttle orbiter environmental control and life support system (ECLSS). ARS was chosen because of its criticality and because it is both complex enough to provide a variety of malfunctions and simple enough to be easily understood by knowledge engineers.

The system maintains a breathable atmosphere, with total pressure about 14.7 psi and partial oxygen pressure between 2.9 and 3.4 psi. The main components of the model are the oxygen and nitrogen supply, check valve, sensors and controller for the partial pressure of oxygen (ppO_2), and the control valve and its actuator. When the ppO_2 exceeds 3.4 psi, it is detected by the ppO_2 sensors, and the ppO_2 controller causes the control valve actuator to switch to the OPEN position. This opens the control valve, and nitrogen flows into the cabin regulator. The nitrogen applies reverse pressure to the check valve, stopping the oxygen flow. Crew breathing depletes the existing oxygen, and eventually the partial pressure drops below 2.9 psi. The controller valve is then closed and the nitrogen flow stops, enabling oxygen to enter the cabin (Figure 3).

The failures that can be detected by the existing prototype are in the ppO_2 sensors and controller, the control valve, the oxygen and cabin regulators, the oxygen pressure sensor, and the cabin pressure sensor.

SAFE Operation

As long as no failure is detected, the user interface displays the sensor values on a user-oriented display. For prototype verification purposes, a software simulator for the ARS was written by using the procedural capabilities of OPS83. The simulator supplies the sensor data. Twelve malfunctions have been incorporated into the simulator, and the demonstrator can choose and inject one through the simulator interface (refer to the simulator section).

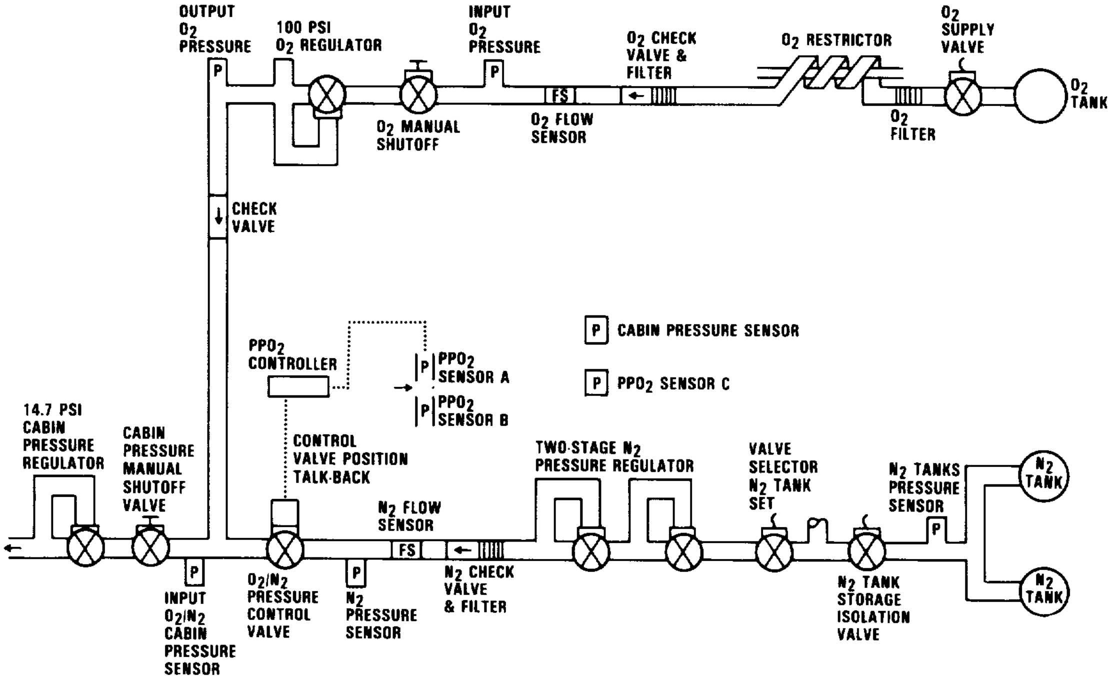
OUTPUT O_2 PRESSURE
100 PSI O_2 REGULATOR
INPUT O_2 PRESSURE
O_2 CHECK VALVE & FILTER
O_2 RESTRICTOR
O_2 SUPPLY VALVE
O_2 TANK
P
P
O_2 MANUAL SHUTOFF
FS
O_2 FLOW SENSOR
O_2 FILTER
CHECK VALVE
PPO2 CONTROLLER
P PPO2 SENSOR A
P PPO2 SENSOR B
P CABIN PRESSURE SENSOR
P PPO2 SENSOR C
14.7 PSI CABIN PRESSURE REGULATOR
CABIN PRESSURE MANUAL SHUTOFF VALVE
CONTROL VALVE POSITION TALK-BACK
N_2 FLOW SENSOR
TWO-STAGE N_2 PRESSURE REGULATOR
VALVE SELECTOR N_2 TANK SET
N_2 TANKS PRESSURE SENSOR
N_2 TANK
N_2 TANK
FS
P
P
P
INPUT O_2/N_2 CABIN PRESSURE SENSOR
O_2/N_2 PRESSURE CONTROL VALVE
N_2 PRESSURE SENSOR
N_2 CHECK VALVE & FILTER
N_2 TANK STORAGE ISOLATION VALVE

Figure 3. ECLSS O_2/N_2 Pressurization System

Once the expert system has detected a malfunction, it notifies the user, instructing him on how to perform a safing procedure to restore the system to an operational state. If, however, the symptoms are insufficient to uniquely identify the failure, the user can choose to run tests to isolate the malfunction further. The appropriate steps for a test procedure are then displayed. The expert system receives information like acknowledgments of actions taken or requests for explanations through the user interface, which runs as a separate process from the expert system. The expert system can continue to monitor sensor data while the user interface is engaged in man-machine dialog.

SAFE Diagnosis Mechanism

A complete and well-defined architecture for a shallow-reasoning diagnosis and safing system now exists. The entire process can be described as a cycle through the phases illustrated in Figure 4. The diagnosis activity is separated into sensor and nonsensor device failure detection phases so that sensor problems can be found before the readings are relied upon to diagnose other failures. The process involves the following steps:

- The sensor diagnosis rules are enabled. If a sensor is determined to be faulty or is suspected of being faulty, it is so marked. This stage can generate a diagnosis for a failed sensor.

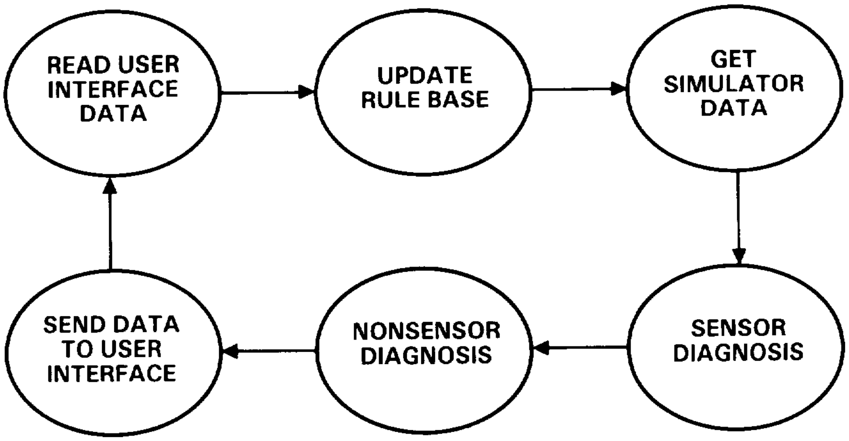

Figure 4. Major Phases of Rule-Based Architecture

- Next, the nonsensor diagnosis rules are enabled. Any rules that base their diagnosis on readings from a sensor previously marked as possibly faulty are not enabled, so no diagnosis is made on the basis of unreliable sensor data. This stage can generate a diagnosis for a failed component.

- A diagnosis can be "final," meaning that the system has made as detailed an isolation process as can be expected, or it can be "intermediate," meaning that tests can narrow it further. The diagnosis then lists all the suspected components and their potential failure mode.

- If the diagnosis is not a final one and the user signals his interest in additional tests, test rules are enabled to create a test procedure. The test procedure is a series of operational instructions to the crew, the purpose of which is to reconfigure the subsystem in such a manner as to permit further isolation of the fault.

- Once the test is completed, the resultant sensor data, coupled with knowledge of the prior diagnosis, enable the system to determine which component really failed.

- If the diagnosis is as specific as possible or if the user chooses not to run any tests, safing rules generate the appropriate safing procedure, which is a series of instructions on how to reconfigure the system to ensure safe vehicle operation, with emphasis given to a safe return home.

- The expert system creates the test and safing procedures dynamically, going from the current system state to the desired one (e.g., it will not instruct to close a closed switch). It also takes into account the flight history (e.g., if the backup system were activated due to a previous failure, there would be no backup system left and the emergency supply system would have to be used).

- The user has to acknowledge the completion of every step of the testing or safing procedure.

- When all safing actions are completed, the system suspends the diagnosis process to permit all sensor values to converge to their nominal ranges. Otherwise, the failure that just occurred might be reannunciated, since typically the sensor values would be out of range for a while. This step is called "system stabilization." The approach is too simplistic; should another failure occur, the user would not know about it until the system is stabilized. Other solutions should be investigated.

- A mechanism for handling certain multiple malfunction situations is introduced. The malfunctions are given two severity ratings. Throughout the diagnosis process, all diagnosis rules for failures of similar or less criticality than that of the current diagnosis (i.e., the diagnosis being handled) are disabled. If a more severe failure is detected, the current diagnosis is replaced by the new one and the user is instructed to take care of that one first. If, after the safing, the "old" failure is still valid, it influences the sensor readings and eventually is rediagnosed by the expert system. In many cases, a switch is made to the backup system, and failure of a component of the now-inactive system does not manifest itself. However, this approach works only for failures whose signatures are totally independent of each other.

The knowledge base also contains a representation of the systems using OPS83 memory elements. Employing this model, the expert system keeps track of the system's state. It is separate from the simulator's model, and only the sensor information is copied from the simulator to that model via the simulator interface. Thus the rule base has no access path to the failure information in the simulator.

User Interface

The user interacts with SAFE and the simulator via a multilevel menu structure. Communication is effected through the selection of menu and support icons using touch-panel technology.

The user is able to monitor the ARS by viewing either a performance display or a schematic display. The performance display provides a high-level picture of the system status by displaying five color-coded dials that show the status of five critical parameters, by the numerical values indicated as well as by the color of the needles as they swing into the various safety levels on the dial faces. Also, several less critical parameters are represented in tabular form.

The user may also view the status of the same parameters in a more hardware-oriented manner by selecting a schematic view of the system. In this mode, the data appear alongside the components of the relevant subsystem (in this prototype, the ARS), allowing the user to view the physical relationships among the various components.

The user can choose any of three system logs by selecting the appropriate icon: the Status Log, which contains messages relevant to the status of the entire system; the Event Log, which logs any change of state of the ARS subsystem; and the Message Log, which records all communications between the system and the user. The logs are time-stamped and, when relevant, color-coded to indicate the severity of the message. Between the graphic displays and the different logs, the user has an opportunity to choose the level of detail that suits his needs. If he is particularly interested in the ARS performance, he can look at the schematic view; but if he is busy with other mission tasks, he may wish to see only the information in the Status Log.

When a failure is detected by the expert system, the icons necessary to handle the situation appear on the screen, a color-coded message appears in the message window at the bottom of the screen, and the DECtalk device annunciates the message. If the message is of low severity, it is displayed in amber and annunciated once. However, if it is of high severity, it is displayed in red and annunciated until the user acknowledges the message. At this point, the user may do one of two things: safe the system or, if possible and time permits, test the system to isolate the cause of the malfunction.

When the decision is made to safe the system or to perform a test before safing, the user is guided every step of the way via the DECtalk, printed messages, and schematic displays of the relevant cockpit panels with highlighted switches. If at any point the user wants to know the motivation for the actions being recommended, an explanation icon is available. The ensuing explanation is printed in the message window but is neither annunciated by the DECtalk nor recorded in the Message Log.

The communication scheme employed by the user interface to interact with the expert system and SI is based on message passing. The first field of each message is a type field that guides the message's interpretation by its receiver. This allows the message format to be flexible and extensible.

Simulator and Simulator Interface

The simulation model, though simplistic, yields results sufficient to provide a good test bed for the prototype. It produces the values of 13 sensors, emulating the gradual changes and fluctuations that characterize flows and pressure measurements.

When a demonstrator chooses to inject a failure into the system, the failure information is transmitted from the user interface to the simulator interface, where it is converted into the relevant simulator parameters. The simulator iterates constantly, creating sensor values and logging them in the knowledge base, where the expert system can monitor them. Once the simulator creates the right signature (which could take a number of cycles, since the simulator gradually converges on the right sensor values), the proper rules generate the failure diagnosis. The expert system and the simulator interface are in constant communication with the user interface via the VAX mailboxes.

Since the simulator is not a real part of the diagnosis system (existing only for testing purposes), it was decided to completely isolate it from the other components. On the other hand, it was desirable to separate the simulator from the details involved in its communication with the rest of SAFE. This approach allows the communication functions to be isolated into a single module—the simulator interface module—that must be rewritten in order to interface with real hardware. The module is responsible for converting relevant user inputs into data the simulator can manipulate and reporting the simulated sensor readings to the expert system. The expert system has no direct access to the simulator, so it is easy to verify that it gets no data it would not receive under real circumstances.

Host Hardware and Software

The knowledge base, simulator, and simulator interface are implemented in OPS83, a language targeted for rule writing. It has good pattern-matching capabilities, allows user customization of the inference engine, and, because it is compiled, runs at a fairly high speed. Having the simulator also written in OPS83, with its procedural capabilities, permits fast and easy interaction between the simulator and the knowledge base.

The user interface is written in Ada. It runs on a Tektronix 4128 high-resolution graphics work station connected to a VAX 11/785. An Elographics touch panel allows the user to communicate with the expert system via an icon-driven menu. This serves as the sole input device. A Digital Equipment Corporation DECtalk voice synthesis unit annunciates all messages that appear on the screen.

SUMMARY AND CONCLUSIONS

The architecture described in this paper could be used to automate the diagnosis and safing process for any subsystem of the orbiter by replacing the safing and diagnosis rules. Given a good and cooperative expert, this would be a relatively straightforward process. It has some basic capabilities for handling complex situations. Faulty sensor detection, done before the nonsensor devices diagnosis, precludes detection based on erroneous data. The rating and failure-interruption scheme is a first attempt to address the issues of multiple malfunctions detection. The creation of dynamic test and safing procedures enables the expert system to save time and minimize errors that could occur when redundant actions are possible. Dynamic information exchange with the user interface gives the system flexibility and room for expansion.

Many aspects of expert system technology application to fault detection, isolation, and recovery require more investigation. A primary area of research is that of cooperative expert systems that monitor all vehicle subsystems and exchange information. Such an architecture must be developed before the complexity of the whole vehicle can be mastered.

Some diagnosis capabilities are still lacking. Among these are better handling of multiple failures, mainly the inclusion of failures whose signatures interfere with each other; refinement of the postsafing monitoring process; intermittent failure diagnosis; and recognition of unexpected anomalous conditions that have not been encoded explicitly, at least to the extent of notifying the user that the vehicle is not operating correctly and that the problem cannot be diagnosed by the expert system.

The SAFE prototype has demonstrated the feasibility and viability of the technology beyond just Space Shuttle applications. The same concepts can be applied to any on-board automated FDIR system. This technology is destined to become a key component in the design of next-generation space vehicles.